PATENT

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re the Application of

Kuzemchak et al. (TI-32443)	Conf. No. 6142
Serial No. 10/017,777	Group Art Unit: 2192
Filed: December 13, 2001	Examiner: Vo

For: Method and Tool for Verification of Algorithms Ported From One Instruction Set Architecture to Another

SECOND SUBSTITUTE APPELLANTS' BRIEF

Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

Dear Sir:

Appellants respectfully present their brief in support of their appeal of the final rejection of claims in this case. The Notice of Appeal was filed on August 3, 2005, as indicated on the date of the automated facsimile receipt from the Patent and Trademark Office. This Substitute Appellant's Brief is presented in response to the Notice of Non-Compliant Appeal Brief mailed August 1, 2006.

Real Party in Interest

The real party in interest in this application is Texas Instruments Incorporated.

Related Appeals and Interferences

Related copending application S.N. 10/022,972 is currently on appeal. Its claim 1 includes method steps similar to method steps that are present, *inter alia*, in claim 10 of this application. This claim 1 is under final rejection, and is on appeal.

Status of the Claims

Claims 1 through 3, 6, 8, 10, 13, and 15 were finally rejected in the Office Action of May 3, 2005, and are the subject of the present appeal.

Claims 4, 5, 7, 9, 11, 12, 14, and 16 stand allowed.

Status of Amendments

An amendment was presented after the final rejection, on August 3, 2005, directed to formal matters regarding claims 10 and 16. This amendment of August 3 has been entered.

A Terminal Disclaimer of this application relative to copending related application S.N. 10/022,972, was filed in this application contemporaneously with the original Appellants' Brief, to remove obviousness-type double patenting as one of the grounds of rejection in this application.

Summary of the Claimed Subject Matter

The subject matter of independent claim 1 is directed to a method for verifying the equivalence of two software programs that are executed by hardware systems that operate under different instruction set architectures. A first program is executed at source hardware (107) with one instruction set architecture,[1] such as under the control of emulation hardware (105), and a second program is executed at target hardware (108) with a different instruction set architecture,[2] which may be under the control of another instance of emulation hardware (107). The claim requires that the instruction set architecture of the source hardware differ from the instruction set

[1] Specification of S.N. 10/017,777, at page 4, lines 23 through 30; Figure 1.
[2] Specification, *supra,* at page 4, line 23 through page 5, line 5; Figure 1.

architecture of the target hardware.[3] The inventive method collects first and second sets of events from the execution of the first and second software programs, respectively,[4] and determines whether the first set of events is equivalent to the second set of events, by reconciling the first and second sets of events with one another, and indicating that the first and second programs are not equivalent if an unreconciled event is discovered.[5] Examples of the reconciling of such events include the identifying of "unreconciled" states generated by either the source or target hardware, such states including writes to an accumulator, address register, or to memory, and a comparing of unreconciled states between the source and target executions.[6]

The subject matter of independent claim 10 is similarly directed to a method of verifying the equivalence of two software programs executed by hardware systems that operate under different instruction set architectures, including substantially the steps of independent claim 1. Claim 10 recites the steps of executing a first program,[7] executing a second program,[8] and collecting first and second sets of events created by the first and second programs, respectively, while executing.[9] The method of independent claim 10 also states that this collecting of the first and second sets of events is performed, at least in part, by calculating (702) a current effective address delay of a given instruction in the instruction pipeline of the corresponding hardware system.[10] For example, this current effective address delay can correspond, for example, to the number of CPU clock cycles between entry of the instruction into the pipeline, on one hand, to the point at which the effective address for the instruction will be computed, on the other hand.[11] The method finds (704, 706) whether a current effective address is available based on that current effective address delay.[12] If a valid effective address is not available, the effective

[3] Specification, *supra,* page 8, lines 6 through 17, Table 1 and Table 2, and Figures 4A through 4H.
[4] Specification, *supra,* page 6, line 15 through page 7, line 7; Figures 2 and 3.
[5] Specification, *supra,* page 7, line 8 through page 8, line 5; Figures 2 and 3.
[6] Specification, *supra,* page 6, line 22 through page 7, line 16.
[7] Specification, *supra*, page 4, lines 23 through 30.
[8] Specification, *supra,* page 4, line 23 through page 5, line 5; Figure 1
[9] Specification, *supra,* page 6, line 15 through page 7, line 7; Figures 2 and 3.
[10] Specification, *supra,* page 18, lines 6 through 15; Figure 7.
[11] Specification, *supra,* page 18, lines 7 through 10.
[12] Specification, *supra,* page 18, lines 16 through 24 (for example, if the current effective address delay for that instruction is negative, the effective address may have been compromised).

address of the instruction is computed (710, 712).[13] In either case, the collecting step reports the effective address of the instruction.[14]

The subject matter of independent claim 15 is directed to a digital software verification system, including a general purpose computer system (100) in combination with software development systems (102) corresponding to source and target hardware. In a system of the type corresponding to independent claim 15, the general purpose computer system directs the execution of the two programs to be verified,[15] according to program instructions corresponding to a verification method substantially as recited in claim 1.

The specification points out numerous important advantages of the methods and systems included in the claimed subject matter. These advantages include the providing of automatic verification of a software application ported to another instruction set architecture by reporting discrepancies at or near the instruction at which the source and target versions vary.[16]

Appellants respectfully submit that the claims on appeal do not include any means-plus-function or step-plus-function elements.[17] As such, no additional identification regarding the structure, material, or acts described in the specification as corresponding to each claimed function is presented.[18]

Grounds of Rejection to Be Reviewed On Appeal

The rejection of claim 1 and its dependent claims

Claim 1 was finally rejected under §102 as anticipated by the Guerra et al. reference[19]. The Examiner asserted that the Guerra et al. reference discloses the executing and collecting

[13] Specification, *supra,* page 18, line 23 through page 19, line 1.
[14] Specification, *supra,* page 19, lines 1 through 6.
[15] Specification, *supra,* page 4, line 23 through page 5, line 5; Figure 1.
[16] Specification, *supra,* page 4, lines 3 through 10; page 21, lines 15 through 25.
[17] 35 U.S.C. §112, ¶6.
[18] 37 C.F.R. §41.67 (c)(1)(v).
[19] Guerra et al., "Cycle and Phase Accurate DSP Modeling and Integration for HW/SW Co-Verification", *ACM* (June, 1999), pp. 964-69.

steps, with reference to ISA and RTL models and a "co-verification" model.[20] The Examiner also asserted that the Guerra et al. reference teaches the determining and indicating steps, by reference to its co-simulation, and waveform and software debugging.[21]

In response to Appellants' subsequent arguments,[22] the Examiner further asserted that the Guerra et al. reference:

> . . . discloses the hardware execution structures, the first simulator as the ISA DSP and second simulator as co-verification. Each model/version is attached to its hardware structure, e.g., ISA mode representing a source hardware is run against RTL model (co-verification) representing target hardware.[23]

The dependent claims were rejected based on additional findings of teachings in the Guerra et al. reference applied against the limitations presented in those dependent claims.

The rejection of claim 10

Claim 10 was finally rejected under §102 as anticipated by the Guerra et al. reference as applied against claim 1, and additionally because of the Examiner's finding of the detailed recitation of the collecting steps to also be disclosed by the reference. More specifically, the Examiner found the Guerra et al. teachings regarding its pipeline and branch prediction meet the claim steps of determining, calculating, finding, computing and reporting, which are involved in finding and reporting effective addresses of instructions in a pipeline.[24]

The rejection of claim 15

Claim 15 was finally rejected under §102 as anticipated by the Guerra et al. reference, based on an assertion that the "VHDL environment, as seen in Pages 964-965, Introduction, and Section 2, Processor Co-verification model, and Figure 5" disclose the structural elements of the first and second microprocessors and the first and second emulation hardware recited in claim

[20] Office Action of May 3, 2005, p. 4.
[21] *Id., citing* Guerra et al., *supra,* Figure 7.
[22] In response to Appellants' arguments in the Amendment Under Rule 116 of August 3, 2005.
[23] Advisory Action of September 7, 2005, continuation sheet..
[24] Office Action, *supra,* page 7, *citing* Guerra et al., *supra,* page 966, section 2.3, Figures 2 and 3.

15.[25] The Examiner also asserted that the Guerra et al. reference teaches that the general purpose computer is programmed to perform the recited method steps, on the same grounds as asserted against claim 1[26].

Argument

It is axiomatic that a finding of anticipation, under §102, is established only if all the elements of the patent claim are identically described, either expressly or inherently, in a single prior art reference.[27]

Claim 1 and its dependent claims

(a) *The reference does not teach executing a first program at source hardware and executing a second program at target hardware*

Claim 1 requires the executing of a first program at source hardware operating according to a first instruction set architecture, and the executing of a second program at target hardware operating according to a second instruction set architecture. Appellants submit that claim 1 and its dependent claims are novel over the Guerra et al. reference because the reference does not disclose source hardware of a first instruction set architecture, and target hardware of a second instruction set architecture, and therefore necessarily fails to disclose the executing steps of claim 1 and its dependent claims.

The Guerra et al. reference is directed solely to a simulation system for modeling and verifying software for a DSP.[28] It is clear from the tenor of Section 2 of the Guerra et al. reference that its method is applied in the modeling, and system simulation, of a particular DSP.[29] It is the results of this model that are analyzed, or compared against the results of other models for that device, in performing the verification method disclosed in the Guerra et al.

[25] Office Action, *supra,* page 7.
[26] Office Action, *supra,* pages 7 and 8.
[27] *Verdegaal Bros. v. Union Oil Co. of California*, 814 F.2d 628, 631, 2 USPQ2d 1051, 1053 (Fed. Cir. 1987); *Richardson v. Suzuki Motor Co.*, 868 F.2d 1226, 1236, 9 USPQ2d 1913, 1920 (Fed. Cir. 1989); MPEP §2131.
[28] Guerra et al., *supra,* page 964, Abstract.
[29] Guerra et al., *supra,* page 965-67, §2 (*i.e.,* §§2.1 through 2.4).

reference.[30] Examples of models compared in the Guerra et al. reference include an instruction set architecture (ISA) model against a gate/RTL model.[31]

However, the Guerra et al. reference nowhere discloses that its method is performed by source and target hardware that execute first and second programs, much less in a manner required by the collecting, determining, and indicating steps of the claim. Instead, the Guerra et al. reference discloses but a single hardware system ("a 296 MHz Sun Ultra2 with 2 Gigabytes of RAM"[32]) that runs both models. There is simply no disclosure whatsoever from the Guerra et al. reference of the executing of a first program at source hardware, and the executing of a second program at target hardware. Models are not hardware, they are instead software; as such, two distinct models are not two distinct hardware systems.

The Examiner asserts that the claims do not require executing of the first and second programs in two different systems.[33] Appellants submit that claim 1 and its dependent claims in fact do require such executing. Specifically, claim 1 requires the step of "executing a first program at source hardware", and the step of "executing a second program at target hardware". The source hardware and target hardware are clearly expressed, in the claim, as distinct from one another.

The Guerra et al. reference falls short of teaching the executing steps of claim 1 and its dependent claims in another respect. Claim 1 clearly requires that the source hardware operates according to a first instruction set architecture, and that the target hardware operates according to a second instruction set architecture. Interpretation of the plain language of this claim would indicate that these first and second instruction set architectures are distinct from one another. However, the Guerra et al. reference refers to only one instruction set architecture (ISA), namely that of the DSP architecture being verified. But the reference is asserted as teaching, and in fact only teaches, the verification of this ISA model against a gate/RTL model (or mixed-level

[30] *See* Guerra et al., *supra,* page 968, §4.1.
[31] Guerra et al., *supra,* page 964, right-hand column, ¶3.
[32] Guerra et al., *supra,* page 968, last paragraph.
[33] Advisory Action, *supra,* continuation sheet.

VHDL model).[34] As known in the art and as evident from the reference itself, the RTL model is a "hardware" model, and does not correspond to an "instruction set architecture". Nowhere does the reference disclose the verification of model results between first and second instruction set architectures (ISAs); the reference only discloses the single ISA model that is verified against the hardware model. As such, the Guerra et al. reference also unambiguously fails to disclose both source hardware operating according to a first instruction set architecture, and target hardware operating according to a second instruction set architecture, as required by claim 1. Accordingly, Appellants submit that the Guerra et al. reference fails to disclose the executing steps of claim 1 on appeal.

(b) The reference does not teach the determining and indicating steps of claim 1

Appellants submit that the Guerra et al. reference falls further short of the requirements of claim 1, in that the reference fails to disclose the determining and indicating steps of claim 1. Specifically, claim 1 requires the additional steps of determining if the first set of events is equivalent to the second set of events by reconciling the first set of events and the second set of events, and of indicating the first program is not equivalent to the second program if an unreconciled event is discovered during the step of determining. Neither of these steps is taught by the reference.

As mentioned above, the Examiner refers to "co-simulation as disclosed by this reference, debugging, showing waveform and software debug of Figure 7"[35] as teaching the determining and indicating steps of the claim. In a previous action, the Examiner asserted that the last four lines of the Abstract of the Guerra et al. reference, and also its section 3.4, teach these steps.[36] However, there was no assertion made by the Examiner of whether or how the reconciling of events of the two sets of events, or the discovering of unreconciled events, are taught by the reference. As such, we are left with examining the cited locations of the reference to determine whether these steps are taught.

[34] Office Action, *supra;* Guerra et al., *supra,* page 964, right-hand column; page 965, left-hand column, first two paragraphs.
[35] Office Action, *supra,* page 5.

The last four lines of the Abstract of the Guerra et al. reference make no reference to reconciling of events, but instead refers to the performance of "[t]he use of co-verification models in place of the RTL".[37] The "Debugging" section 3.4 of the reference merely provides a general discussion of debugging, *e.g.,* "breakpoints can be placed within the co-verification model to step, break, and perform all other standard software debug operations".[38] And the cited Figure 7 of the reference is merely a screen shot "showing waveform and software debug" – it is far from apparent how this portion of the reference in any way approaches the determining and indicating steps of claim 1.

As noted above, it is axiomatic that a finding that a claim is anticipated, under §102, is proper only if all of the claim elements are identically described, either expressly or inherently, by the reference.[39] The Examiner has failed to provide any specific teaching in the reference of the determining and indicating steps of claim 1, but instead has only asserted that debugging, in a general sense, meets these steps. Appellants submit that the finding by the Examiner falls short of that required under the law for a proper anticipation rejection, and also submit that the Guerra et al. reference in fact does not teach the determining and indicating steps of claim 1.

(c) The rejection of claim 1 is therefore in error

For these reasons, Applicants submit that the final rejection of claim 1 and its dependent claims is in error, and that these claims are in fact novel over the Guerra et al. reference. Reversal of the final rejection of claims 1 through 3, 6, 8, and 13 is therefore respectfully requested.

[36] Office Action of August 26, 2004, page 3.
[37] Guerra et al., *supra,* Abstract.
[38] Guerra et al., *supra,* page 968, section 3.4.
[39] *Verdegaal Bros., supra; Richardson, supra*; MPEP §2131.

Claim 10

(a) The reference does not teach the determining and indicating steps of claim 10

Similarly as claim 1, discussed above, claim 10 requires the determining of whether a first collected set of events created from the executing of a first program is equivalent to a second collected set of events created from the executing of a second program, and the indicating that the programs are not equivalent if an unreconciled event is discovered during this determining step.

And for the same reasons as discussed above relative to claim 1, Appellants submit that the Guerra et al. reference fails to disclose these determining and indicating steps. As discussed above, the Examiner provides no assertion of whether or how the reconciling of events of the two sets of events, or the discovering of unreconciled events, are taught by the reference, but merely cites certain locations of the references in connection with "debugging", or "calculating", or other generalized activity. And also as discussed above, the portions of the reference that were cited by the Examiner as teaching the determining and indicating steps, in fact do not teach those steps. Nor does the remainder of the reference disclose these steps.

Appellants therefore submit that the finding by the Examiner falls short of that required under the law for a proper anticipation rejection, and also submit that the Guerra et al. reference in fact does not teach the determining and indicating steps of claim 10.

(b) The reference does not teach the specific steps comprising the collecting steps as required by claim 10

As mentioned above, claim 10 further recites specific steps that comprise the steps of collecting a first set of events and collecting a second set of events. These specific steps include calculating the current effective address delay of an instruction in the pipeline, finding that a valid effective address is available based on that current effective address delay, computing the effective address if a valid effective address is not available, and reporting the effective address.

The Examiner refers to the handling of conditional branch instructions[40] in the reference as teaching these additional steps of claim 10. Appellants respectfully submit that this portion of the reference has no relation to these steps of claim 10. That portion of the reference simply refers to the delaying of interrupt handling if a conditional branch resides in the processor pipeline at the time of an interrupt.[41] More specifically, this portion of the reference discloses that, in the event of an interrupt occurring with a conditional branch in the instruction fetch pipeline stage, the interrupt is delayed by one instruction cycle so that a "delay slot" instruction is executed before the interrupt is serviced, permitting the target address of the branch to be used as the return address from the interrupt routine.[42] However, the Guerra et al. reference in no way teaches the calculating of any effective address delay of an instruction, nor the determining of whether a valid effective address for that instruction is available based on that effective address delay, nor the computing of an effective address if a valid effective address is not available, all such steps required by claim 10. This complete lack of correspondence of the Guerra et al. reference to these steps of claim 10 is further evident from the absence of any specific assertions by the Examiner that the reference teaches the features of "current effective address delay", "valid effective address", "computing the effective address", all as present in claim 10.

Appellants submit that the finding by the Examiner falls short of that required under the law for a proper anticipation rejection, and also submit that the Guerra et al. reference in fact does not teach these steps of claim 10.

(c) The rejection of claim 10 is therefore in error

For these reasons, Applicants submit that the final rejection of claim 10 is in error, and that this claim is novel over the Guerra et al. reference. Reversal of the final rejection of claim 10 is therefore respectfully requested.

[40] Guerra et al., *supra,* page 966, right-hand column, section 2.3; Figures 2 and 3.
[41] *Id.*
[42] Guerra et al., *supra,* page 966, right-hand column, last paragraph.

Claim 15

(a) The reference does not teach first and second microprocessors and first and second emulation hardware as required by claim 15

Claim 15 is directed to a digital software verification system that comprises a general purpose computer, in combination with first and second microprocessors that are each for executing application programs. The claimed system further requires first and second emulation hardware, coupled between the general purpose computer and the first and second microprocessors, respectively. The claim further recites that the first emulation hardware controls the operation of the first microprocessor according to a first instruction set architecture, while the second emulation hardware controls the operation of the second microprocessor according to a second instruction set architecture. The general purpose computer is further recited as programmed to perform a method for verifying equivalence of a target application program to a source application program.

No specific assertion was made by the Examiner regarding whether these specific structural elements are taught by the Guerra et al. reference. Rather, the Examiner asserted that the "VHDL environment, as seen in Pages 964-965, Introduction, and Section 2, Processor Co-verification model, and Figure 5" disclose these structural elements.[43] However, it is clear from these portions of the reference, and indeed from the reference in its entirety, that there is no disclosure of the combination of a general purpose computer with first and second microprocessors, and first and second emulation hardware as claimed. Rather, the Guerra et al. reference discloses but a single hardware system ("a 296 MHz Sun Ultra2 with 2 Gigabytes of RAM"[44]) that runs both of its models. There is simply no disclosure whatsoever from the Guerra et al. reference a first microprocessor and a second microprocessor, or of first and second emulation hardware, in combination with its Sun Ultra2 system.

The Examiner later asserted that the reference teaches "hardware execution structures, the first simulator as the ISA DSP and second simulator as co-verification", referring to the models

[43] Office Action, *supra,* page 7.
[44] Guerra et al., *supra,* page 968, last paragraph.

executed by the Guerra et al. system.[45] Appellants submit, however, that these simulators are models, and not microprocessors. A simulator of a microprocessor is not a microprocessor. Models are not hardware, but are software. As such, two distinct models are not two distinct hardware systems. And therefore, the two models of the Guerra et al. reference do not teach the first and second microprocessors of the claim.

For this reason, Appellants submit that claim 15 is novel over the Guerra et al. reference, because at least four specific structures required by the claim are not taught by the reference.

(b) The reference does not teach the first and second instruction set architectures, according to which the first and second emulation hardware respectively control the first and second microprocessors, as claimed

As discussed above, the Guerra et al. reference is directed solely to a simulation system for modeling and verifying software for a particular DSP,[46] with the results of this model that are analyzed, or compared against the results of other models, in performing the verification method disclosed in the Guerra et al. reference.[47] Examples of models compared in the Guerra et al. reference include an instruction set architecture (ISA) model against a gate/RTL model.[48]

Claim 15 also requires, as mentioned above, that the first emulation hardware control the operation of the first microprocessor according to a first instruction set architecture, and that the second emulation hardware control the operation of the second microprocessor according to a second instruction set architecture. Appellants submit that the Guerra et al. reference also fails to disclose these limitations. Specifically, the Guerra et al. reference refers to only one instruction set architecture (ISA), namely that of the DSP architecture, which is being verified against a gate/RTL model (or mixed-level VHDL model).[49] As known in the art and as evident from the reference itself, the RTL model does not correspond to an "instruction set architecture". Accordingly, the reference fails to disclose the verification of model results between first and

[45] Advisory Action, *supra*.
[46] Guerra et al., *supra,* page 965-67, §2 (*i.e.,* §§2.1 through 2.4).
[47] *See* Guerra et al., *supra,* page 968, §4.1.
[48] Guerra et al., *supra,* page 964, right-hand column, ¶3.

second instruction set architectures (ISAs), much less disclose first and second emulation hardware that control operation according to these first and second instruction set architectures, respectively.

Appellants therefore submit that the Guerra et al. reference fails to disclose the first emulation hardware and the second emulation hardware of claim 15. And therefore, Appellants submit that the Guerra et al. reference fails to disclose a general purpose computer that is programmed to perform a method comprising the steps of causing the first emulation hardware to execute a first program at the first microprocessor, and of causing the second emulation hardware to execute a second program at the second microprocessor.

(c) The rejection of claim 15 is therefore in error

For these reasons, Applicants submit that the final rejection of claim 15 is in error, and that the claim is novel over the Guerra et al. reference. Reversal of the final rejection of claim15 is therefore respectfully requested.

* * * * *

[49] Office Action, *supra;* Guerra et al., *supra,* page 964, right-hand column; page 965, left-hand column, first two paragraphs.

For the foregoing reasons, Appellants respectfully submit that the final rejection under §102 of claims 1 through 3, 6, 8, 10, 13, and 15, as anticipated by the Guerra et al. reference, is in error. Reversal of the final rejection of the claims in this case is therefore respectfully requested.

Respectfully submitted,

/rma/

Rodney M. Anderson
Registry No. 31,939
Attorney for Appellants

Anderson, Levine & Lintel, L.L.P.
14785 Preston Road, Suite 650
Dallas, Texas 75254
(972) 664-9554

Claims appendix:

1. A method for verifying that two programs are equivalent, the method comprising the steps of:

executing a first program at source hardware operating according to a first instruction set architecture;

collecting a first set of events that are created by the first program while it is being executed;

executing a second program at target hardware operating according to a second instruction set architecture;

collecting a second set of events that are created by the second program while it is being executed;

determining if the first set of events is equivalent to the second set of events by reconciling the first set of events and the second set of events; and

indicating the first program is not equivalent to the second program if an unreconciled event is discovered during the step of determining.

2. The method of Claim 1, wherein the step of determining comprises the steps of:

matching each event of the first set of events to a corresponding event in the second set of events;

declaring the existence of an unreconciled event if an event in the first set of events does not have a corresponding event in the second set of events; and declaring the existence of an unreconciled event if an event in the second set of events does not have a corresponding event in the first set of events.

3. The method of Claim 2, wherein:

the step of collecting a first set of events comprises developing a logical state for each event of the first set of events;

the step of collecting a second set of events comprises developing a logical state for each event of the second set of events; and

the step of matching comprises comparing the logical state of each event of the first set of events to the logical state of the corresponding event in the second set of events.

6. The method of Claim 1, wherein the step of indicating comprises:

activating a debugging capability in a software development system during the step of executing a first program, at the point where the unreconciled event is discovered; and

activating the debugging capability in the software development system during the step of executing a second program, at the point where the unreconciled event is discovered.

8. The method of Claim 1, wherein:

the steps of collecting a first set of events and collecting a second set of events comprise treating references to address registers in instructions as symbols; and the step of determining comprises computing the effective addresses of the analogous references in the first set of event and the second set of events when reconciling the first set of events and the second set of events.

10. A method for verifying that two programs are equivalent, the method comprising the steps of:

executing a first program;

collecting a first set of events that are created by the first program while it is being executed;

executing a second program;

collecting a second set of events that are created by the second program while it is being executed;

determining if the first set of events is equivalent to the second set of events by reconciling the first set of events and the second set of events; and

indicating the first program is not equivalent to the second program if an unreconciled event is discovered during the step of determining;

wherein the steps of collecting a first set of events and collecting a second set of events comprise:

determining that a first instruction is in the instruction pipeline;

calculating the current effective address delay of the instruction in the pipeline;

finding that a valid effective address for the instruction is available based on the current effective address delay of the instruction;

computing the effective address of the first instruction if a valid effective address is not available; and

reporting the effective address of the instruction.

13. The method of Claim 1, wherein the means for executing, collecting, determining, and indicating is a first verification program whereby the verification program causes a first program to be executed by a first software development system, causes a second program to be executed by a second software development system, and receives information from the first software development system and the second software development system to perform the steps of collecting, determining, and indicating.

15. A digital software verification system, comprising:

a general purpose computer;

a first microprocessor for executing application programs;

first emulation hardware, coupled between the first microprocessor and the general purpose computer, for controlling the operation of the first microprocessor according to a first instruction set architecture;

a second microprocessor for executing application programs; and

second emulation hardware, coupled between the second microprocessor and the general purpose computer, for controlling the operation of the second microprocessor according to a second instruction set architecture;

wherein the general purpose computer is programmed to perform a method for verifying that a target application program is equivalent to a source application program, the method comprising the steps of:

causing the first emulation hardware to execute a first program at the first microprocessor;

collecting a first set of events that are performed by the first program while it is being executed;

causing the second emulation hardware to execute a second program at the second microprocessor;

collecting a second set of events that are performed by the second program while it is being executed;

determining if the first set of events is equivalent to the second set of events; and

indicating the first program is not equivalent to the second program if a mismatch is discovered during the step of determining.

Evidence appendix:

None.

Related proceedings appendix:

None.